

Portland State University PDXScholar

Electrical and Computer Engineering Faculty
Publications and Presentations

Electrical and Computer Engineering

1999

TRACE: A Visual Software System to Explore Properties of Reed-Muller Movement Functions

Marek Perkowski

Portland State University, marek.perkowski@pdx.edu

Alan Mishchenko

Portland State University

Let us know how access to this document benefits you.

Follow this and additional works at: http://pdxscholar.library.pdx.edu/ece_fac

 Part of the [Electrical and Computer Engineering Commons](#)

Citation Details

Perkowski, Marek and Mishchenko, Alan, "TRACE: A Visual Software System to Explore Properties of Reed-Muller Movement Functions" (1999). *Electrical and Computer Engineering Faculty Publications and Presentations*. Paper 225.
http://pdxscholar.library.pdx.edu/ece_fac/225

This Conference Proceeding is brought to you for free and open access. It has been accepted for inclusion in Electrical and Computer Engineering Faculty Publications and Presentations by an authorized administrator of PDXScholar. For more information, please contact pdxscholar@pdx.edu.

TRACE: A Visual Software System to Explore Properties of Reed-Muller Movement Functions

Alan Mishchenko and Marek A. Perkowski

Dept. of Electrical and Computer Engineering, Portland State University,
P.O. Box 751, Portland, OR 97207-0751, USA, [alanmi, mperkows]@ee.pdx.edu

Abstract

We present new experimental Windows 95/98/NT software for investigation of graph properties of boolean (in particular, Reed-Muller) logic with an equal number n of inputs and outputs (called movement functions). Realized at the input of an n -bit register, such functions create autonomous Finite State Machines (FSMs). TRACE software system allows the user to visualize State Transition Graphs (STGs) of the autonomous FSMs. Other features of TRACE help explore graph properties of function families. These families are produced by a generic function, differing from it only in the order of components, one operation, or one literal (this literal is complemented or replaced by another literal). The autonomous FSMs are used to implement economically next-state logic of real-time control units such as CPU controllers. A case study using TRACE to build economical, highly testable reversible counters based on linear Reed-Muller polynomials is given.

Introduction

The classical approach to FSM synthesis is based on finding an acceptable state encoding and then deriving boolean equations for flip-flop excitation signals and outputs. In practice, another approach is often used. This approach relies on the use of counters or shift registers for the automatic generation of code sequences. The states are encoded by superimposing the counter or shift register state sequences over state sequences in the STG of the FSM. In this case, the implementation of the FSM is based on the embedded counter or shift register.

Here we consider a generalization of the approach from [2]. The basic idea is that we can use not only a counter or a shift register but an *arbitrary* autonomous FSM (FSM without inputs) as the device to be embedded into the FSM under design. In particular, we study properties and implementation of autonomous FSMs created by linear Reed-Muller polynomials [1].

Movement Functions and Autonomous Finite State Machines

The autonomous FSM is implemented using a register with a feedback consisting of combinational logic (Figure 1) [3]. The register can include flip-flops a_j of any type, but first we will limit our attention to D flip-flops. Let us denote the direct (complemented) flip-flop outputs as a_j (respectively, \bar{a}_j) and the feedback logic outputs (or the flip-flop inputs) as f_n, \dots, f_2, f_1 . The feedback logic consists of a set of boolean functions

$$f_j = f_j(a_n, a_{n-1}, \dots, a_2, a_1). \quad (1)$$

Functions f_j may have any form, in particular, they can be the constants 1 or 0. Simple functions, such as EXORs of a few literals, are of special interest.

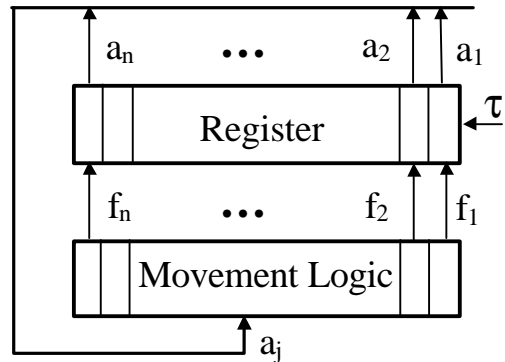


Figure 1. Structure of the autonomous FSM.

An ordered set of n such functions

$$F = \{f_n, f_{n-1}, \dots, f_2, f_1\} \quad (2)$$

is called the *movement function* of the autonomous FSM. For brevity, we will include only the parts on the right side of equations (1) into set (2). For instance, the function describing a cyclic shift by one bit to the left is specified as follows

$$F_{shift} = \{a_{n-1}, a_{n-2}, \dots, a_2, a_1, a_n\}. \quad (3)$$

Polynomial counters are based on the following functions

$$F_{polyn} = \{a_{n-1}, a_{n-2}, \dots, a_2, a_1, a_n \oplus a_r\}, \quad (4)$$

where the symbol \oplus stands for the EXOR operation, and integer r depends on the number of flip-flops n , as shown in Table 1.

Due to feedback, a code $Q = \{q_n, q_{n-1}, \dots, q_2, q_1\}$ stored in the D flip-flop register of the autonomous FSM is transformed into a new code $Q' = \{q_n', q_{n-1}', \dots, q_2', q_1'\}$ in the next clocking period. Each bit of code Q is transformed according to the equation

$$q_j' = f_j(Q),$$

and each state Q of the register is succeeded by the next state Q' determined from the formula

$$Q' = F(Q).$$

In particular, for some transitions, the code Q' can be the same as the code Q .

Given a movement function F , all 2^n possible codes stored in the register are arranged in a sequence. We call this sequence *the trace of the movement function F* . For example, the trace of n -bit cyclic shift function (3) consists of a number of n -code cycles plus one or more cycles of a shorter length (in Figure 2, this trace is shown for $n = 4$). The trace of polynomial function (4) includes a zero code cycle and a long cycle of the remaining $2^n - 1$ codes.

Graph Theoretic Properties of STGs of Movement Functions

Traces of movement functions have the following properties:

- Any vertex has only one vertex-follower (in other words, trace graphs have no fan-out branchings).
- Any vertex can have from 0 up to 2^n vertices-predecessors (in other words, a vertex is a starting vertex or it has a fan-in branching that is limited by the total number of vertices). For example, if the movement function consists of k constants 1, all the vertices of its trace converge in the vertex, whose code is "1" repeated k times.
- The trace graph may have from 1 (as in the modulo-2 counter) up to 2^n (as in the movement function of parallel transfer, where $f_j = a_j$) isolated parts, without transitions between them.
- In each isolated part of the graph, there is a cycle of length from 1 up to 2^n (see examples from the previous property).
- A vertex belonging to a cycle can be the meeting point of one or more linear code sequences without branching or other more complex vertex structures.
- The movement function, whose k components are described by equations $f_j = a_j$ while other $n-k$ components do not depend on the respective k flip-flop outputs (a_j), has a trace consisting of 2^k identical "pages". The codes on these pages differ only in the "page number" (the part of the code composed of k bits with $f_j = a_j$).

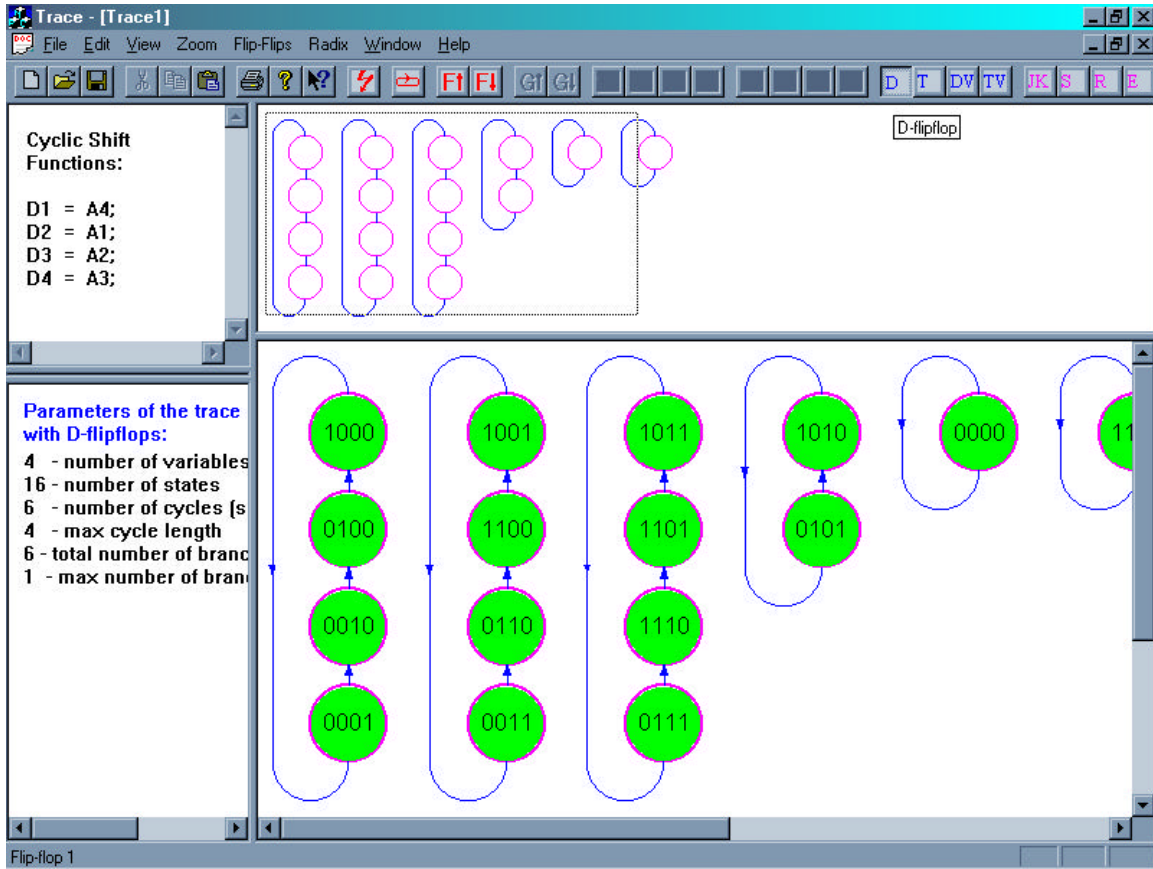


Figure 2. TRACE software.

Overview of TRACE

TRACE is a software package for visualization of the trace graphs of autonomous FSMs and exploring their properties. The form of the trace graphs depends not only on functions themselves but also on the type of flip-flops used in the register (Figure 1).

The system allows the user to enter functions, select the type of flip-flops, and display the resulting graph, which is shown as the top right window. It is also possible to zoom to the part of this graph as shown in the bottom right window (Figure 2). Next the system allows the user to change the function automatically by shifting its components or replacing one literal and observing how the trace of the function changes in response. This quality makes TRACE an excellent tool for researching movement functions.

The system works reliably for a number of inputs (outputs) of movement function less than 16. The following case study illustrates the use of TRACE in research of the properties of Reed-Muller polynomials.

Case Study: Economic Reed-Muller Implementation of Reversible Counters

Theoretical Foundations

Let us consider linear Reed-Muller polynomials used in polynomial counters and synthesis of arbitrary automata based on polynomial counters.

Definition 1. An n -bit boolean function $F = (f_1, f_2, \dots, f_n)$ over n variables $f_1 = f_1(x_1, x_2, \dots, x_n), f_2 = f_2(x_1, x_2, \dots, x_n), \dots, f_n = f_n(x_1, x_2, \dots, x_n)$ is called a linear polynomial if

$$f_j(x_1, x_2, \dots, x_n) = \sum_{i \in Q_j} x_i, \quad (5)$$

where $1 \leq j \leq n$, Q_j is a set of different integers from 1 to n and the sum is an EXOR.

Definition 2. Let $F = (f_1, f_2, \dots, f_n)$ be an n -bit Boolean function of n variables

$$f_1 = f_1(x_1, x_2, \dots, x_n), f_2 = f_2(x_1, x_2, \dots, x_n), \dots, f_n = f_n(x_1, x_2, \dots, x_n).$$

Then the reversible function for F is an n -bit function $G(g_1, g_2, \dots, g_n)$ over n variables such that for every tuple (x_1, x_2, \dots, x_n) of argument values of the function F such that

$$(f_1, f_2, \dots, f_n) = F(x_1, x_2, \dots, x_n),$$

the following equality holds

$$(x_1, x_2, \dots, x_n) = G(f_1, f_2, \dots, f_n).$$

Theorem 1. Given is an n -bit linear polynomial Boolean function $F = (f_1, f_2, \dots, f_n)$ over n variables of the form

$$f_1 = x_n \oplus \sum_{k \in Q} x_k, f_2 = x_1, f_3 = x_2, \dots, f_n = x_{n-1}, \quad (6)$$

Set Q is a collection of m ($0 \leq m \leq n-1$) different integers k that satisfy the inequality $1 \leq k \leq n-1$. In particular, the set Q may be empty. Then for every m , function F has a unique reversible function G , which is defined by the relationship:

$$g_1 = x_2, g_2 = x_3, \dots, g_{n-1} = x_m, g_n = x_1 \oplus \sum_{k \in Q} x_{k+1}. \quad (7)$$

Proof of the theorem [4] is based on solving the matrix equation

$$Ax = f, \text{ where } A = \begin{matrix} & 1 & k_1 & \dots & k_m & n \\ \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{matrix}$$

in Galois field and getting the solution in the form

$$x = A^{-1}f, \text{ where } A^{-1} = \begin{matrix} & 1 & k_{i+1} & \dots & k_{m+1} & n \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \end{matrix}$$

Theorem 2. Let $F = (f_1, f_2, \dots, f_n)$ be an n -bit linear polynomial Boolean function over n variables

$$f_1 = x_1 \oplus x_n \oplus \sum_{k \in Q} x_k, f_2 = x_2 \oplus x_1, f_3 = x_3 \oplus x_2, \dots, f_n = x_n \oplus x_{n-1}, \quad (8)$$

where set Q is a collection of m ($0 \leq m \leq n-2$) different integers k that satisfy the inequality $2 \leq k \leq n-1$. Then for an even m the function F does not have a reversible function, and for an odd m the function F has a unique reversible function G , which is defined by the relationship

$$g_j = \left(\sum_{i=1}^{k_1} x_i \oplus \sum_{i=k_2+1}^{k_3} x_i \oplus \dots \oplus \sum_{i=k_m+1}^{k_m} x_i \right) \oplus \sum_{i=j+1}^n x_i, 1 \leq j \leq n. \quad (9)$$

Practical Applications

Let us now consider applications of these results. Discrete devices often include n -bit binary counters mod 2^n to count the number of input signals in natural code. The feedback function realized on the inputs of the flip-flops used in these counters is a counting function that increments the counter contents by 1 each time the count signal arrives. Reversible counters are also used. These counters, in addition to the counting function, realize a function that subtracts 1 from the counter contents. A reversible counter operates in the two modes: counting up and counting down.

Circuits that realize the counting functions are relatively complex and slow because counting depends on producing a carry-over bit and letting it ripple through the register from the lower bits to the higher. Polynomial counters are free from these shortcomings, because they are based on polynomial movement functions (4). The usefulness of polynomial counters is limited, however, because the sequence of binary codes generated in the counter differs from the increasing sequence of n -bit binary numbers in natural code.

The complexity of the circuit realizing polynomial feedback functions depends on the types of flip-flops used as memory elements and on the number of bits in the counter. Thus, given D or SR flip-flops, the polynomial counter is usually implemented by functions (6). The sequence of counter states in this case is a cycle of length 2^n-1 , which does not include only the zero-code (it forms a separate cycle of length 1).

According to Theorem 1, reversible functions of a simple form (7) exist for polynomial functions (6). By Definition 2, when function G , reversible with respect to the original polynomial function F , is implemented in the counter, it produces a cycle of length 2^n-1 . This cycle is formed by the same codes as the previously mentioned cycle and differs from the latter only in that the sequence of codes has the inverse direction.

Thus, implementation of polynomial functions (6) and their reverse functions (7) produces a reversible polynomial counter. The counter circuit in this case is simpler and faster than the ordinary natural code reversible counters based on the counting up and counting down functions. This implementation exists for SR and D flip-flops (and other flip-flops functioning in the SR or D mode).

Given T flip-flops (and other flip-flops functioning in the T mode), the realization of the function (6) on their inputs produces output signals described by function (7). The latter function is the composition of the characteristic function of T flip-flops and function (6). By Theorem 2, given odd number n , reversible functions (9) also exist for polynomial functions (8). Due to the complexity of these functions, their implementation in reversible counters does not meet hardware and speed requirements. In reversible counters with T flip-flops, it is preferable to use the counting up and counting down functions rather than polynomial functions.

Experimental Results

In Table 1 below, the first column contains the number of bits in the counter, the second and third columns contain formulas for the EXOR sum S of additional variables in expressions (6) and (8) for the first component of the polynomial counter found using TRACE:

$$S = \sum_{k \in Q} x_k$$

In the fourth columns, the literal count for the gate-level implementation of reversible counters based on D/SR flip-flops is given. (Given T flip-flops, according to Theorem 2, similar implementations do not exist for any of the counters in Table 1.) Calculation of the literal count LC in the last column of the table is performed according to the formula

$$LC = LC_{up} + LC_{down} + 6*n,$$

where LC_{up} and LC_{down} are literal counts in (6) and (7), respectively. Additional $6*n$ literals in this formula correspond to $3*n$ 2-input gates needed to control counting in different directions. In Figure 3, the circuit of the reversible polynomial counter for $n = 3$, designed using the above method, is given.

Figure 3. Gate-level implementation of the reversible polynomial counter for $n = 3$.

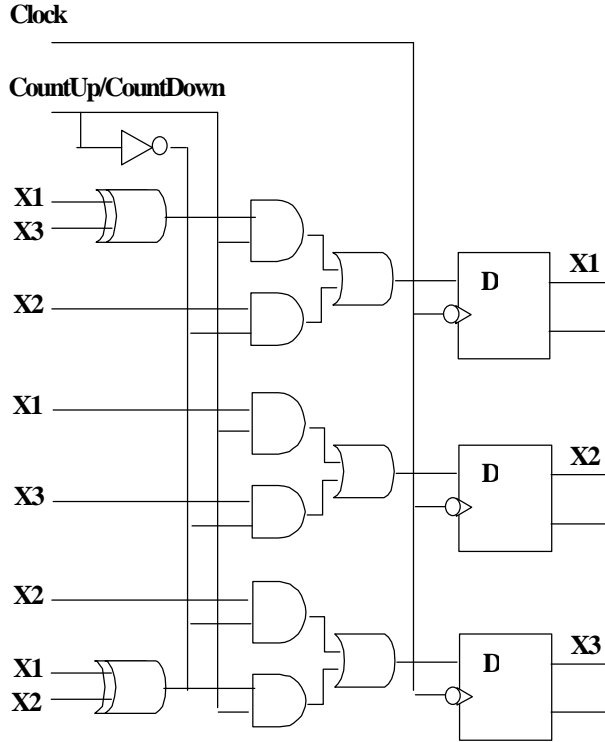


Table 1. The results of design of reversible counters

Number of bits, n	The additional term S in the 1st component		Literal count, LC
	D ffs	T ffs	
2	x_1	x_1	16
3	x_1	x_1	22
4	x_1	x_3	28
5	x_2	x_3	34
6	x_1	x_1	40
7	x_1	x_1	46
8	$x_1 \oplus x_3 \oplus x_5$	$x_1 \oplus x_3 \oplus x_5$	56
9	x_4	x_4	58
10	x_3	x_3	64
11	x_2	x_2	70
12	$x_1 \oplus x_4 \oplus x_6$	x_7	80
13	$x_3 \oplus x_5 \oplus x_{10}$	$x_3 \oplus x_5 \oplus x_{10}$	86
14	$x_1 \oplus x_4 \oplus x_8$	$x_1 \oplus x_4 \oplus x_8$	92
15	x_1	x_4	94

Conclusion

Experimental results show that TRACE can be used by hardware designers looking for a good fit of the FSM under design and the autonomous FSM based on a movement function. By going over a number of similar functions that comprise a family in the sense described above, an economic implementation of the real-life FSM may be found with the help of TRACE.

As the case studies show, another possible use of TRACE is finding gate-level implementations of economical, highly testable reversible counters based on Reed-Muller polynomials. Such counters and arbitrary FSMs based on these counters can be obtained when the outputs of polynomial functions (6) and their reversible functions (7), controlled by the signal CountUp/CountDown, are fed into the inputs of SR and D flip-flops. For T flip-flops, there is no simple implementation of reversible counters.

Due to the attractive visual qualities of graphs created by TRACE, it can be used in university education. The authors have successfully incorporated it into logic design classes to demonstrate the properties of movement functions to electrical engineering students.

TRACE is available on the web [5].

References

- [1] A. Gill. *Linear Sequential Circuits: Analysis, Synthesis, and Applications*. McGraw-Hill, 1966.
- [2] A. T. Mishchenko. A control unit synthesis method. *Kibernetika*, 3, 1972, pp. 148-149 (in Russian).
- [3] Yu. V. Kapitonova, A. T. Mishchenko. Logic design of universal automata. *Kibernetika*, 5, 1986, pp. 32-46 (part 1); 6, 1986, pp. 44-57 (part 2) (in Russian).
- [4] A. A. Mishchenko. On properties of reversible polynomial counters. *Kibernetika i sistemny analiz*, 5, 1997, pp. 44-49 (in Russian).
- [5] <http://www.ee.pdx.edu/~alanmi/software/index.htm>